文章编号：2096-1472(2023)-01-46-04
DOI:10.19644/j.cnki.issn2096-1472.2023.001.010

融合变异测试的路径覆盖测试数据进化生成方法

党向盈，李金凤

(徐州工程学院信息工程学院，江苏 徐州 221018)
✉dangpaper@163.com; 41407770@qq.com

摘　要：针对检测缺陷的测试数据生成效率低下问题，提出变异测试和路径覆盖测试技术结合的测试数据生成方法。首先，采用变异测试技术生成的变异分支融入程序，生成新的被测程序；然后，在原路径集中挑选目标路径，通过分析变异分支与路径关联关系，将变异分支融入路径。最后，基于遗传算法生成覆盖路径的测试数据。实验结果表明，多种群遗传算法生成测试数据的时间，比单种群遗传算法节约了41.15%。由此可见，对于覆盖多路径测试数据生成，多种群遗传算法的效率比单种群遗传算法高。

关键词：软件测试；变异测试；测试数据生成；遗传算法
中图分类号：TP311　**文献标识码**：A

An Evolutionary Generation Method for Path Coverage Test Data based on Mutation Testing

DANG Xiangying, LI Jinfeng

(*School of Information Engineering*, *Xuzhou University of Technology*, *Xuzhou* 221018, *China*)
✉dangpaper@163.com; 41407770@qq.com

Abstract: Aiming at the low efficiency of test data generation for defect detection, this paper proposes a test data generation method combining mutation testing and path coverage testing. Firstly, the mutation branches generated by mutation testing technology are incorporated into the original program to generate a new program under test. Then, some paths of the original program are selected as the target paths, and the mutation branches are integrated into the target path by the correlation between mutation branches and paths. Finally, Test results show that the time of generating test data by Multi Population Genetic Algorithm (MGA) is 41.15% less than that by Single Population Genetic Algorithm (SGA). It can be seen that the efficiency of MGA is higher than that of SGA for test data generation of multi-path coverage.

Keywords: software testing; mutation testing; test data generation; GA

1 引言(Introduction)

软件测试是保障软件产品质量的一种有效方式。随着科学技术的飞速发展，软件产品的复杂性越来越高，规模也越来越大，这也导致软件缺陷越来越隐蔽，采用传统的检测方法很难检测出软件的顽固缺陷[1]。实施自动化软件测试技术时，需要测试数据执行程序，同时测试数据是检验程序缺陷的重要手段，因此测试数据的质量尤为重要[2]。

变异测试是一种面向缺陷的软件测试技术，它通过对程序语句合乎语法的改变，模拟真实的缺陷[3]。通过这种方式，对模拟的缺陷生成测试数据，保障测试数据集的充分度。在

基金项目：江苏省高等学校自然科学研究重大项目(21KJA520006)；徐州市科技计划项目(KC21007).

结构覆盖测试技术中，覆盖率最高的是路径覆盖测试技术。路径中包含的节点是程序的语句，所以覆盖一条路径的测试数据能覆盖很多程序语句，那么路径覆盖所得的测试集规模也比较小，冗余测试数据少。

本文借鉴变异测试和路径覆盖技术生成能检测缺陷的测试数据。变异测试模拟缺陷，融合原程序路径，覆盖路径生成的测试数据，也能覆盖缺陷，而且生成的测试数据集规模比较小。

2　测试数据生成方法(The proposed method of test data generation)

结构化的路径覆盖测试所生成的用例是为了覆盖目标路径，即使路径覆盖率高，也不能代表检测缺陷率高。此外，传统的方法生成测试数据效率比较低，考虑到进化算法已经广泛应用[4]，本文采用遗传生成测试数据覆盖融合缺陷的路径。

变异测试和路径覆盖测试技术结合的测试数据生成方法的基本框架如图1所示。首先基于变异测试技术生成变异体，并转化为变异分支，用于模拟程序的缺陷，再将这些变异分支(缺陷)融入程序；同时，在原程序路径集中选择多条目标路径。然后，将变异分支融入原程序生成新被测程序；通过分析变异分支与原语句之前的相关性，将变异分支融入目标路径。最后，基于遗传算法生成覆盖路径的测试数据。

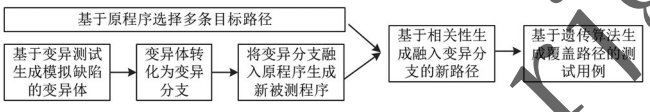


图1 变异测试和路径覆盖测试技术结合的测试数据生成方法的基本框架

Fig.1 Basic framework of test data generation method based on mutation testing and path coverage testing

2.1　基于变异测试技术模拟程序缺陷

变异测试技术不仅可以根据程序或语句的特征模拟真实软件中的各种类型的缺陷，也可以基于程序的复杂情况，针对性地选择缺陷发生位置和注入缺陷的数目[5]。因此，变异测试被认为是一种方便、灵活、个性化的技术。

在实施变异测试时，通过变异算子对原程序语句实施变异操作，比如原语句为“a>b”,实施变异之后，生成变异语句“a==b”。将变异语句替换原来的语句，得到新的被测程序定义为变异体。每一个模拟的缺陷对应一个变异体。在实施变异测试时，一个程序通常会产生大量的变异体。某一测试数据同时执行原程序和变异体，如果两者的输出结果不一致，那么该用例以强变异测试准则杀死变异体。而且，为了杀死这些变异体，需要大量的测试数据执行变异体和原程序，导致变异测试代价非常高昂[6]。

为了克服以上不足，文献[6]提出弱变异测试思想，将变异体转化为变异分支。具体方法是根据变异测试的必要条件，将原语句和变异语句构建if条件语句的分支，形成变异分支。如果某一测试数据覆盖了变异分支，那么基于弱变异测试准则杀死变异体。图2(a)为三角形类型判断程序的部分源代码，如果对语句n_3(“a>c”)实施变异，得到变异语句“a≥c”。然后，基于文献[6]提出的方法进行弱变异测试转化，构建变异分支“if (a>c)!=(a>=c)…”；采用同样的弱变异测试方法，对语句n_7、n_{11}实施变异，得到图2(b)中的16 个变异分支，分别表示为$e_1,e_2,\cdots,e_{16}$。将这些变异分支插入原语句n_3的前面，得到新的被测程序。

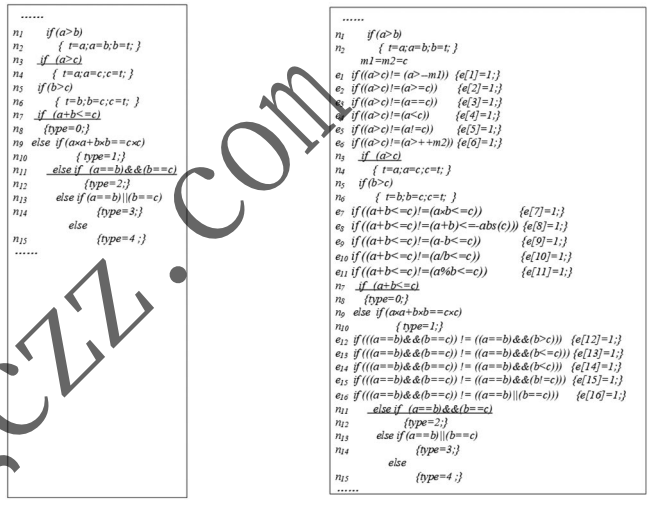


图2 示例程序

Fig.2 Sample program

插入原程序的变异分支，代表一些模拟缺陷。如果某一测试数据覆盖某一变异分支，说明基于弱变异测试准则杀死了对应的变异体，也就是说，生成的测试数据可以用于检测这个模拟的缺陷。

2.2　基于相关性变异分支融入路径

变异分支插入原程序后，生成新的程序。考虑到原程序中可以通过路径覆盖生成测试数据，对于新程序，为了生成覆盖变异分支的测试数据，可以将这些变异分支融入已有原程序路径，这样，生成的测试数据不仅满足路径覆盖准则，也满足变异测试准则。

为了将变异分支融入原程序路径，首先，在原程序路径集中选出若干目标路径；然后，在新程序中判断原语句与变异分支的相关性，生成相关子路径；最后，基于原语句所在目标路径将变异分支插入目标路径，形成新的目标路径。以图2为例阐述基于相关性变异分支融入路径的方法。在原程序路径集中选择目标路径，可以得到：

$$\overline{p_1}=n_1,n_3,n_4,n_5,n_6,n_7,n_9,n_{11},n_{13},n_{14}$$

$\overline{p_2} = n_1, n_3, n_5, n_7, n_9, n_{11}, n_{13}, n_{14}$

$\overline{p_3} = n_1, n_3, n_5, n_7, n_9, n_{11}, n_{12}$

需要说明的是，路径集中可能包含很多路径，针对挑选目标路径的原则：(1)尽量选择包含路径节点比较少的路径。(2)挑选的目标只是初选，如果所有变异分支都无法融入，该路径将被删除。(3)目标路径中，还有未融入的变异分支，将从路径集中重新挑选路径。

本文借鉴文献[7]提出的方法，判断变异分支与原语句的相关性，并生成相关子路径。具体方法如下：设n_i为被测程序原语句，n_i的真分支表示为$n_i(1)$，n_i的假分支表示为$n_i(0)$。如果n_i为条件语句的谓词表达式，那么考察e_j的条件语句与n_i之间的相关性：如果它们是真真相关的[8]，那么基于变异分支和被测语句的真分支形成一条子路径，记为$(e_j, n_i, n_i(1))$；如果它们是真假相关的，那么基于变异分支和被测语句的假分支形成一条子路径，记为$(e_j, n_i, n_i(0))$。如果被测语句不是条件语句的谓词表达式，那么直接基于变异分支与该被测语句形成一条子路径，记为(e_j, n_i)。如图2示例中，变异分支e_1与原语句n_3是真假相关，因此形成一条子路径(e_1, n_3, n_5)。再比如，变异分支e_6与原语句n_3是真真相关，因此形成一条子路径(e_6, n_3, n_4)；接着，将变异分支与原语句形成的相关性子路径融入原程序的目标路径。图2(b)中的16 条变异分支融入目标路径后的新路径如下：

$p_1 = n_1, e_6, n_3, n_4, n_5, e_7, e_8, n_6, n_7, n_9, n_{11}, n_{13}, n_{14}$

$p_2 = n_1, e_4, e_5, n_3, n_5, e_9, e_{10}, e_{11}, n_7, n_9, e_{13}, e_{14}, e_{15}, e_{16}, n_{11}, n_{13}, n_{14}$

$p_3 = n_1, e_1, e_2, e_3, n_3, n_5, n_7, n_9, e_{12}, n_{11}, n_{12}$

2.3 基于遗传算法生成测试数据

遗传算法是由自然界的遗传和进化理论启发而来，是一种被广泛使用的全局搜索的优化技术。近年来，遗传算法已经广泛应用于软件测试中[9]。

设目标路径为p_i，某一进化个体穿越的路径为$\tilde{p}$，设目标路径和穿越路径的相似度可以表示为

$$f(X) = \frac{|p_i \cap \tilde{p}|}{\max(|p_i|, |\tilde{p}|)}$$

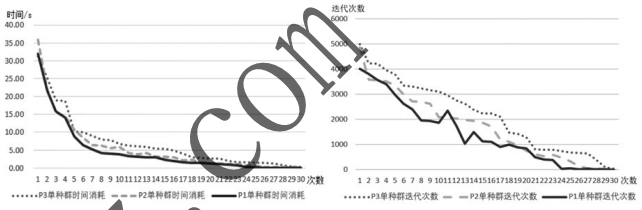
式中，$|p_i|$和$|\tilde{p}|$为p_i和$\tilde{p}$的节点数目。$|p_i \cap \tilde{p}|$为路径p_i和$\tilde{p}$从前往后相同节点的数目。式中路径的相似度[9]作为遗传算法中适应值函数$f(X)$，用于评价进化个体的优劣。

本文采用遗传算法时，遗传操作包括轮盘赌选择方法，即单点交叉和单点变异。交叉概率和变异概率分别为0.9和0.3。考虑到目标路径有多条，本文采用单种群遗传算法(Single Population Genetic Algorithm，SGA)和多种群遗传算法(Multi-population Genetic Algorithm，MGA)生成覆盖路径测试数据。单种群遗传算法生成测试数据时，一次运行算法只能针对一条路径生成测试数据。多种群遗传算法生成测试数据时，一次运行算法，多个种群针对多条路径生成测试数据，多种群个数设置为目标路径个数。

3 实验 (Experiments)

图2示例程序生成的三条目标路径(p_1, p_2, p_3)，分别采用单种群遗传算法和多种群遗传算法生成测试数据。为了消除算法执行的随机性，在生成测试数据时，算法独立执行30 次，然后计算它们的平均值。评价指标选择测试数据生成的时间消耗和迭代次数。实验中，单种群遗传算法和多种群遗传算法分别执行30 次，对每次执行时间消耗和迭代次数进行记录，并按照降序排列。

采用单种群遗传算法分别执行30 次，覆盖三条目标路径(p_1, p_2, p_3)的时间消耗和迭代次数对比如图3所示。从图3中可以看出，采用单种群遗传算法生成测试数据时，在时间消耗方面，覆盖路径p_3时间消耗最大，覆盖路径p_1时间消耗最小；在迭代次数方面，p_3迭代次数最大，p_1迭代次数最小。

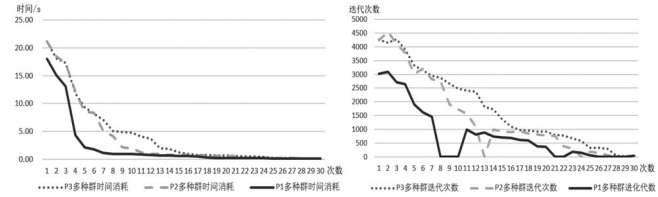


(a)时间消耗 (b)迭代次数

图3 三条目标路径(p_1, p_2, p_3)采用单种群遗传算法的时间消耗和迭代次数对比

Fig.3 Comparison of time consumption and iteration number of three target paths (p_1, p_2, p_3) using SGA

如图4所示，三条目标路径(p_1, p_2, p_3)采用多种群遗传算法的时间消耗和迭代次数对比结果。由该图可以看出，覆盖p_1成本最小，覆盖p_3成本最大，路径p_3覆盖难度最大，p_1覆盖难度最小，覆盖难度越大，则生成测试数据的成本越大。



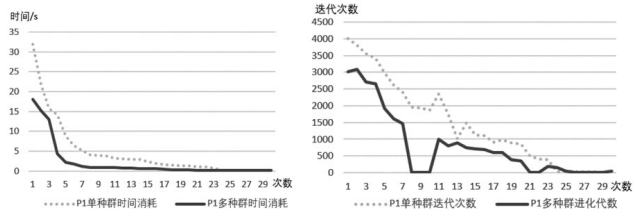
(a)时间消耗 (b)迭代次数

图4 三条目标路径(p_1, p_2, p_3)采用多种群遗传算法的时间消耗和迭代次数对比

Fig.4 Comparison of time consumption and iteration number of three target paths (p_1, p_2, p_3) using MGA

下面比较采用单种群和多种群生成测试数据的性能。由图5所知，对于路径p_1，在时间消耗和迭代次数方面，因为实验具有一定的偶然性，所以单种群遗传算法偶尔优于多种群遗传算法，但从总体来看，多种群遗传算法生成测试数据方法明显优于单种群遗传算法。同理，由图6可知，对于路径p_2，多种群遗传算法在时间消耗和迭代次数方面的性能也高于单种群遗传算法。由图7可知，对于路径p_3，多种群遗传算法的性能也

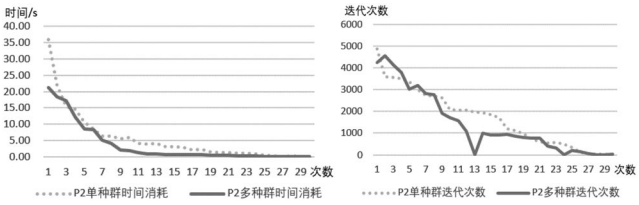
高于单种群遗传算法。



(a)时间消耗　　(b)迭代次数

图5 采用单种群和多种群覆盖p_1的时间消耗和迭代次数对比

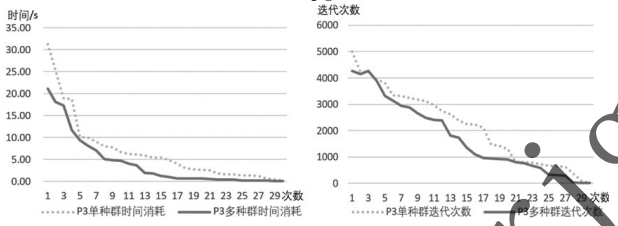
Fig.5 Comparison of time consumption and iteration number of covering p_1 using SGA and MGA



(a)时间消耗　　(b)迭代次数

图6 采用单种群和多种群覆盖p_2时间消耗和迭代次数对比

Fig.6 Comparison of time consumption and iteration number of covering p_2 using SGA and MGA



(a)时间消耗　　(b)迭代次数

图7 采用单种群和多种群覆盖p_3时间消耗和迭代次数对比

Fig.7 Comparison of time consumption and iteration number of covering p_3 using SGA and MGA

表1 单种群和多种群遗传算法时间消耗和迭代次数的平均值

Tab.1 Average of time consumption and iteration numbers using SGA and MGA

路径	时间消耗均值/s		迭代次数均值	
	单种群	多种群	单种群	多种群
p_1	4.72	2.17	1,415.77	793.30
p_2	5.50	3.63	1,680.50	1,429.10
p_3	6.84	4.24	2,148.63	1,746.30
总计	17.06	10.04	5,244.90	3,968.70

对于单种群遗传算法和多种群遗传算法，执行30次的时间消耗和迭代次数平均值如表1所示。从表1中可以看出，覆盖所有路径时，单种群比多种群遗传算法消耗的时间更长，迭代次数更多。多种群遗传算法在时间消耗方面比单种群遗传算法少用7.02 s，时间节约幅度为41.15%。多种群遗传算法在迭代次数方面比单种群遗传算法少1,276.2次，迭代次数降低24.33%。此外，对比覆盖三条目标路径(p_1,p_2,p_3)的平均时间消耗和迭代次数可知，p_3消耗时间最多，迭代次数也最多，说明路径p_3最难覆盖；p_1消耗时间最少，迭代次数也最少，说明路径p_1最容易覆盖。

4 结论(Conclusion)

本文所提出的基于变异测试的路径覆盖测试数据进化生成方法，创新性地将变异测试技术和路径覆盖测试技术结合，转化后的变异分支模仿缺陷，融入原程序路径，生成的测试数据不仅规模小且能够用于检测缺陷。考虑到遗传算法有利于提高测试数据生成效率，针对多条路径，采用多种群遗传算法，每个子种群负责一条路径测试数据的生成。实验结果也验证多种群遗传算法明显优于单种群遗传算法。

参考文献(References)

[1] 蔡开元,孙昌爱,聂长海.软件可靠性评估的控制论观点[J].中国科学:信息科学,2019,49(11):1528-1531.

[2] YAO X, GONG D, ZHANG G. Constrained multi-objective test data generation based on set evolution[J]. IET Software, 2015, 9(4):103-108.

[3] PAPADAKIS M, KINTIS M, ZHANG J, et al. Mutation testing advances: an analysis and survey[J]. Advances in Computers, 2019, 112(2):275-378.

[4] DANG X Y, GONG D W, YAO X J. Enhancement of Mutation Testing via Fuzzy Clustering and Multi-population Genetic Algorithm[J]. IEEE Transactions on Software Engineering, 2022, 23(4):2426-2463.

[5] NISHTHA J, BHARTI S, SHWETA R. Systematic Literature Review on Search Based Mutation Testing[J]. e-Informatica Software Engineering, 2017, 11(1):61-78.

[6] PAPADAKIS M, MALEVRIS N. Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing[J]. Software Quality, 2011, 19(4):691-723.

[7] YAO X, GONG D. Automatic detection of infeasible paths in software testing[J]. IET software, 2010, 4(5):361-370.

[8] 党向盈,巩敦卫,姚香娟.基于统计分析的弱变异测试可执行路径生成[J].计算机学报,2016,39(11):2355-2370.

[9] YAO X, GONG D. Genetic algorithm-based test data generation for multiple paths via individual sharing[J]. Computational Intelligence and Neuroscience, 2014, 29(1): 1-13.

作者简介：

党向盈(1978-)，女，博士，副教授.研究领域：软件测试，进化算法应用.

李金凤(1980-)，女，硕士，实验师.研究领域：计算机应用，软件开发.